Developing an Agent Systems Reference Architecture

Duc N. Nguyen[1], Robert N. Lass[1], Kyle Usbeck[1], William M. Mongan[1], Christopher T. Cannon[1], William C. Regli[1], Israel Mayk[2] and Todd Urness[2]

[1] Applied Communications and Information Networking Institute, Drexel University
{dn53, urlass, kfu22, wmm24, ctc82, regli}@cs.drexel.edu
[2] Communications-Electronics Research, Development and Engineering Center, US Army

Abstract. One reason for the slow adoption in industry of agent-oriented methodologies as a paradigm for developing systems is the lack of integration and general-purpose technologies. To this end, there is a need to define common patterns, relationships between components, and structural qualities of an agent system. A reference architecture for agent-based systems would suit this need. This work describes the methodology for constructing an agent systems reference architecture by combining reverse software engineering techniques and tools and a documentation methodology. The goal of the resulting reference architecture is to identify common patterns and relationships between concepts present in agent systems to aid in describing and designing new agent systems.

1 Introduction

Using agent-based approaches for constructing large complex distributed systems can provide advantages over traditional methods. Unfortunately, industry has been slow to adopt this agent-oriented paradigm. One reason for this slow adoption is the lack of integration and general-purpose technologies [7]. Standards bodies such as the Foundation for Intelligent Physical Agents (FIPA)[3] are leading efforts to standardize protocols and formats of an agent-based system. However, there is a need to construct a reference *architecture* that defines the relationships between standardized terms and concepts of an agent-based system. Furthermore, such an architecture would give a set of architectural blueprints and best practices to aid in developing new agent frameworks and systems. To this end, a reference architecture for agent-based systems would speed other standardization efforts and adoption as a viable systems engineering perspective.

The purpose of the Agent Systems Reference Architecture (ASRA) is to describe relationships and structural qualities to support the construction of an agent-based system. The ASRA is created from multiple cross-cutting levels: the framework level, the system behavior level, and agent systems in the context of larger external systems. Constructing such an architecture for general systems is impossible given the various kinds of agent systems. One approach for building a reference architecture would be to analyze the tools used to build agent systems to determine the interactions between the functional concepts of an agent system.

This paper focuses on the process of developing a reference architecture for agent-based systems. Our approach is the application of a modified 4+1 View Model [3] to

[3] http://www.fipa.org

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
MAY 2010		**00-00-2010 to 00-00-2010**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Developing an Agent Systems Reference Architecture	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Drexel University,Applied Communications and Information Networking Institute,Philadelphia,PA,19104**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

See also ADA546004. Eleventh International Workshop on Agent Oriented Software Engineering (AOSE 2010) In conjunction with AAMAS 2010 (Workshop 8) Held in Toronto, Canada on May 10-14, 2010. Sponsored by EOARD.

14. ABSTRACT

One reason for the slow adoption in industry of agent-oriented methodologies as a paradigm for developing systems is the lack of integration and generalpurpose technologies. To this end, there is a need to define common patterns, relationships between components, and structural qualities of an agent system. A reference architecture for agent-based systems would suit this need. This work describes the methodology for constructing an agent systems reference architecture by combining reverse software engineering techniques and tools and a documentation methodology. The goal of the resulting reference architecture is to identify common patterns and relationships between concepts present in agent systems to aid in describing and designing new agent systems.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**6**	

existing agent framework implementations creating five architectural views. We apply this process to functional concepts in an agent system to obtain the reference architecture.

We analyzed three existing agent framework implementations: Cougaar, JADE, and AGLOBE. This methodology is applied to agent frameworks rather than deployed agent systems because the functional concepts defined in the Agent Systems Reference Model (ASRM) [5] are already implemented in these frameworks.

The main contribution of this paper is a methodology for creating an agent systems reference architecture through the application of reverse engineering methodologies combined with the modified 4+1 View model used for documenting existing agent frameworks.

The rest of this paper is organized as follows: The next section provides a summary of related efforts in reference architectures for agent-based systems, and defines the terms *architecture*, *reference architecture* in the context of agent systems and agent frameworks. Section 3 describes the Agent Systems Reference *Model* and its basis for creating the ASRA. Section 4 describes how the 4+1 Model is applied to agent frameworks. Section 5 demonstrates the application of the process to create a portion of the ASRA. Finally, we conclude with a roadmap of continuing work for developing a reference architecture.

2 Background and Related Work

There is no general consensus for the definition of a reference architecture; however, we describe related standards efforts and reference architectures in this section.

The Foundation for Intelligent Physical Agents There are efforts to describe the behavior and interaction of agent systems. The FIPA Abstract Architecture Specification [2]discusses agent system architecture in an effort to promote interoperability and reusability. FIPA intends to provide a generic view on agent systems and describe how specific functionality should interact. This specification states low-level details such as the mechanisms for how agents perform service look-ups. The ASRA also focuses on identifying architectural paradigms and patterns in agent frameworks. So the ASRA acknowledges that service look-ups may be implemented in different ways and identifies the different ways of performing this action.

Reference Architecture for Situated Multiagent Systems Weyns *et al.* [6] developed a reference architecture for situated multiagent systems consisting from an agent and an application environment viewpoints. This architecture was developed through an interative process of analysis and validation studying different agent-based systems. In their reference architecture, the authors constructed multiple documents from different views: the module decomposition, the shared data, and the communicating processes views.

This reference architecture for situated multiagent systems has a similar result of constructing multiple documents for each view; however, the approach is different. The ASRA uses a combination of static and dynamic code analysis of representative agent framework implementations , whereas the former reference architecture examines several multiagent systems implementations.

RCS and RCS Related Work The Real-time Control System, first developed by Barbera, *et al.*at the National Institute of Standards and Technology, is a reference architecturefor hierarchical intelligent control. The RCS reference architecture provides for intelligent control what the ASRA provides for agent systems. As stated by the RCS reference architecture [1]: "the evolution of the RCS concept has been driven by an effort to include the best properties and capabilities of most, if not all, of the intelligent control systems currently known in the literature, from subsumption to SOAR, from blackboards to object-oriented programming."

The notion of a reference architecture has different meanings based on the viewpoints and concerns of the stakeholders. In this work, a reference architecture for agent-based systems is defined as a set of documents addressing patterns and component relationships of the functional concepts set forth in the ASRM.

3 The Agent Systems Reference Model

The Agent Systems Reference Model (ASRM) [5] is a model for software systems composed of agents. It establishes terms, concepts and definitions needed for the comparison of agent systems. The ASRA is an elaboration of the ASRM since it establishes relationships between concepts in agent frameworks and defines structural patterns for those concepts.

The ASRM defines an intelligent agent—or simply agent—as *situated* computational processes that embody one or more of the following qualities: autonomy, proactivity, interactivity, continuous, sociality, and/or mobility. The ASRM also formalizes concepts and layers of organization in an agent-based system. The layers described are: *agents*, *frameworks*, *platforms*, *hosts*, and *environments*. An agent-based system is the set of frameworks, the agents that execute in them, the platform (other software) that supports them and the hosts (hardware) upon which they execute.

The functional concepts of an agent system support overall system execution. They are essential in the definition of the ASRA and are made up of the following: Agent Administration, Security and Survivability, Mobility, Conflict Management, Messaging, Logging, and Directory Services.

4 Methodology for creating an Agent Systems Reference Architecture

Our approach to constructing a reference architecture for agent systems is to create multiple architecture documents by analyzing existing open source agent framework implementations and applying a rigorous 4+1 view model augmented with reverse engineering data.

Deriving 4+1 Views Using Reverse Engineering The 4+1 View Model [3] creates different architectural descriptions, or *views*, of software systems for different interested parties (*e.g.*, system developers, business-persons, customers). Each view identifies and describes the relationships between components and concepts. Interested parties will

view these relationships with different weights and significance, in some cases they are meaningless. The Views of the 4+1 model, and their construction for the ASRA, are described in detail in Section 5.

Our approach in deriving and documenting each architecture is a modified version of the 4+1 approach. Here, we begin by iterating over the functional concepts of the ASRM, and developing the Scenario View consisting of use cases. For each use case, we developed an agent that exercised the scenario, and performed reverse-engineering runtime analysis to obtain the Process View. This data provides a slice of the program (and, thus, the framework and other libraries), through which we can focus our static analysis in the Development View. Finally, this is abstracted into components using clustering algorithms to form the basis of the Logical View. By contrast to traditional 4+1 approaches, we document the most abstract views first and augment each with reverse engineering data or domain knowledge to create the next view.

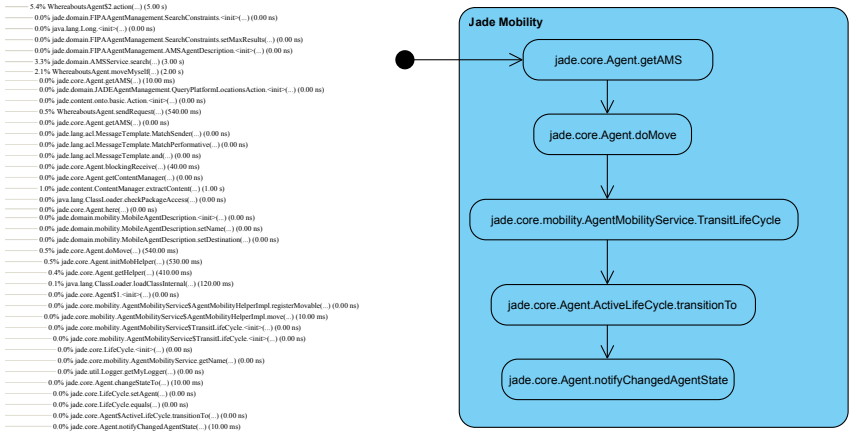
We utilized reverse software engineering tools and performed dynamic and static analysis [4] to assist with our data mining and move from scenarios and existing implementations to a reference 4+1 architectural description. In this approach, the data used to construct a view is used to inform the construction of the next view.

5 Case Study: JADE Mobility

We demonstrate the analysis and application of the modified 4+1 documentation model for agent mobility within an agent system implemented using the JADE framework.

The Scenario View: The documentation process begins with stating the use cases for the functional concept defined by the ASRM and further elaborating actors and invocation points. The intended audience are high-level practitioners who need explanation of concepts for an agent-based system.

The Process View: To create the Process view, dynamic analysis data is generated for a system by running a slice of a program representing the scenario. We create a process diagram to illustrate the process view. Reverse engineered runtime data augments the process diagram to create a package diagram to provide a conceptual view. For agent mobility, we generate a runtime trace by running a profiler against code snippets that demonstrate agent mobility. The runtime trace documents the percentage of time methods are invoked during the code's execution. Figure 1(a) displays the temporal view of a scenario demonstrating the invocation points of the agent mobility functional concept. The resulting process diagram after augmenting with package names is shown in Figure 1(b). Upon creating similar diagrams for AGLOBE and Cougaar, two patterns for defining the process of agent mobility emerge: *serialization mobility* and *shared-object mobility*. With serialization mobility as exhibited by JADE and AGLOBE, (1) the agent's thread of execution is paused, (2) the agent is converted into a transferable form, (3) the is transfered to the target platform, (4) the agent is converted back into an executable form, and (5) the agent resumes (or begins) thread of execution. With shared-object mobility (as is the case with Cougaar) mobile agents are shared between platform containers. The agent's state includes current platform location and



(a) Runtime Trace. (b) Process View Diagram

Fig. 1. Jade Mobility runtime trace and resulting Process view diagram.

upon "transmission", the agent's shared object state is modified to indicate the agent's new platform location.

The Development View: The development view is the static view of the agent system derived through the use of static code analysis tools and temporal data from the previously created process view. Static analysis tools produce a graph of all the software components. This data is analyzed and informed by the runtime analysis (to focus the search of a large static analysis data set) obtained during the construction of the Process View to create the Development View. This view provides a sense of the topology of the agent system software, including the logical components responsible for performing tasks and design patterns representing the connections between them. The resulting agent mobility functional concept is described using the data obtained in the serial approach, and consists of: **Destination Platform Discovery**, in which Directory Services are used to locate the addressing information of the destination platform, **Agent Encapsulation**, in which the mobile agent is serialized into a message from the Messaging functional concept, **Message Communication**, in which the agent is delivered to its destination via the messaging component, and **Agent Extraction**, in which the agent is extracted at the destination platform from its serialized state.

The Logical View: The logical view is constructed by observing the clustered runtime data generated from our reverse engineering tools, and organizing the major objects into packages. Although the process and logical views concern themselves with concrete system details in an architecture, they are also helpful to express the high level packages and interacting components existing in an agent system. For agent mobility, the abstracted logical view illustrates that even though agent mobility can be implemented using the functional concepts for directory services and messaging, there is the additional requirement to identify the messages as encapsulated serialized agents.

The Physical View: The physical view is normally reserved by the 4+1 model for non-functional requirements of a system, including deployment and administration concerns and is developed independently of the serial approach that generated the scenario, process, development, and logical views because it deals with more of the physical aspects of the framework and the agent system.. Although the ASRA addresses some of these concerns, the physical view section primarily concerns itself with the physical aspects and design decisions in deploying an agent system in a given environment. Otherwise this functional concept is not abstractable, so it is omitted here.

6 Conclusion and Future Work

This paper described our approach for creating a reference architecture for agent systems and frameworks using the 4+1 View Model. We demonstrated this application on the Agent Mobility functional concept in each view. This approach satisfies practitioners at multiple levels: high-level, system designer, system architect, developer, system deployer. The ASRA provides architectural design paradigms for agent framework functional concepts defined by the ASRM. These serve as architectural blueprints for constructing new agent frameworks, or identifying the functional components required to construct new systems using existing frameworks. Further documentation of the functional concepts and their interactions at each view is in progress and a complete document is forthcoming. Further work also includes creating reference architectures focusing on the paradigms of agents and external systems outside the traditional agent-based system construct (for example, agents integrated with web services) and on agent societies and communities.

References

1. James Albus and G. Rippey. RCS: a reference model architecture for intelligent control. In *Proceedings of the From Perception to Action Conference*, pages 218—229, September 1994.
2. Foundation for Intelligent Physical Agents. Abstract architecture, December 2002. `http://www.fipa.org/specs/fipa00001/`.
3. P. Kruchten. Architectural blueprints—The "4+1" view model of software architecture. *IEEE Software*, 12(6):42–50, November 1995.
4. W. M. Mongan, C. J. Dugan, R. N. Lass, A. K. Hight, J. Salvage, W. C. Regli, and P. J. Modi. Dynamic analysis of agent frameworks in support of a multiagent systems reference model. *IADIS International Conference Intelligent Systems and Agents*, 2007.
5. W. C. Regli, I. Mayk, C. J. Dugan, J. B. Kopena, R. N. Lass, P. J. Modi, W. M. Mongan, J. K. Salvage, and E. A. Sultanik. Development and specification of a reference model for agent-based systems. *IEEE Trans. On Systems, Man, and Cybernetics, Part C*, 39(5):572–596, Sep. 2009.
6. D. Weyns and T. Holvoet. A reference architecture for situated multiagent systems. *Lecture Notes in Computer Science*, 4389:1, 2007.
7. D. Weyns, H. V. D. Parunak, and O. Shehory. The future of software engineering and multi-agent systems. *Special issue on Future of Software Engineering and Multi-Agent Systems, International Journal of Agent-Oriented Software Engineering (IJAOSE)*, 2008.